andrena OBJECTS

Experts in agile software engineering

# Wie können Softwaremetriken tatsächlich das Entwicklerleben verbessern?

Dr. Andreas Arnold (andreas.arnold@andrena.de)
Dr. Joachim Melcher (joachim.melcher@andrena.de)

Entwicklertag Karlsruhe, 9. Mai 2012

# Metriken - Der erste Ansatz

- Idee: Software-Metriken könnten für Projekt nützlich sein
- Umsetzung:
  - viele Metriken werden automatisiert berechnet
  - Unsicherheit, was damit nun anzufangen ist
  - keine abgeleiteten Handlungen
- Konsequenz: Metriken werden nach kurzer Zeit ignoriert

# Was sind Software-Metriken?

- statische Code-Analyse
- Grundlage Quelltext (bzw. abgeleiteter Kontroll-/Datenflussgraph)
- Quantifizierung einer **internen** Eigenschaft anhand konkreter Regeln

- eigentlich Interesse an **externen** Eigenschaften
- Software-Metrik als **Indikator** für externe Eigenschaft
- Abhängigkeit muss durch Experiment gezeigt werden (in der Praxis oft vernachlässigt!)

# Was können Metriken?

- Vergleich über die Zeit oder mit anderen Projekten
- Indikator für Entwicklungstendenz bzw. aktuellen Zustand
- Hinweis auf mögliche Problemstellen
- Feststellen von Handlungsbedarf
- aber auch: Argumentationsgrundlage für Entwickler gegenüber Management

# Es gibt viele Metriken

## Auswahl einiger Metriken

- Lines of Code (LOC)
- zyklomatische Komplexität (McCabe)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOC)
- Response For a Class (RFC)
- Weighted Methods per Class (WMC)
- Coupling Between Objects (CBO)
- Lack of Cohesion in Methods (LCOM)

# Es gibt viele Metriken

## Mögliche Klassifikation

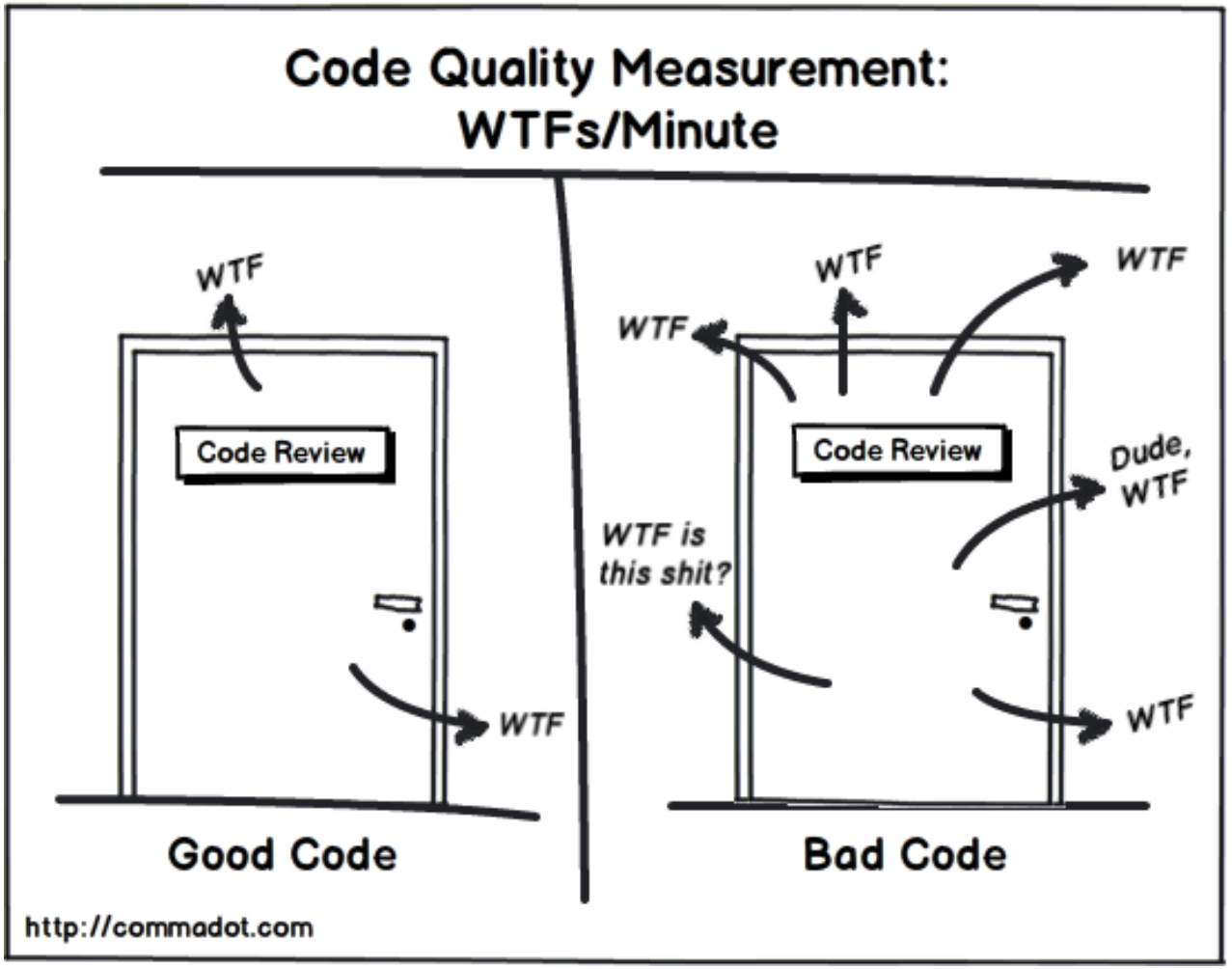
- traditionell
  - Größenmetriken
  - Komplexitätsmetriken
- objekt-orientiert
  - Methodenebene
  - Klassenebene
  - Vererbungshierarchie

# Entscheidungskriterien

- Einfachheit (einfach zu messen/verstehen)
- Handlung ableitbar (einfach zu reparieren)
- Robustheit (Zusammenhang zwischen Metriken)

# Sinnvolle Metriken

## WTFs/min



# Sinnvolle Metriken

## ISIS/USUS-Metriken

- Testabdeckung
- Anzahl Pakete in Zyklen
- Average Component Dependency
- Anzahl Methoden mit zyklomatische Zahl größer 5
- Anzahl Klassen mit mehr als 20 Methoden
- Anzahl Methoden mit mehr als 15 LOC
- Anzahl Compiler-Warnungen

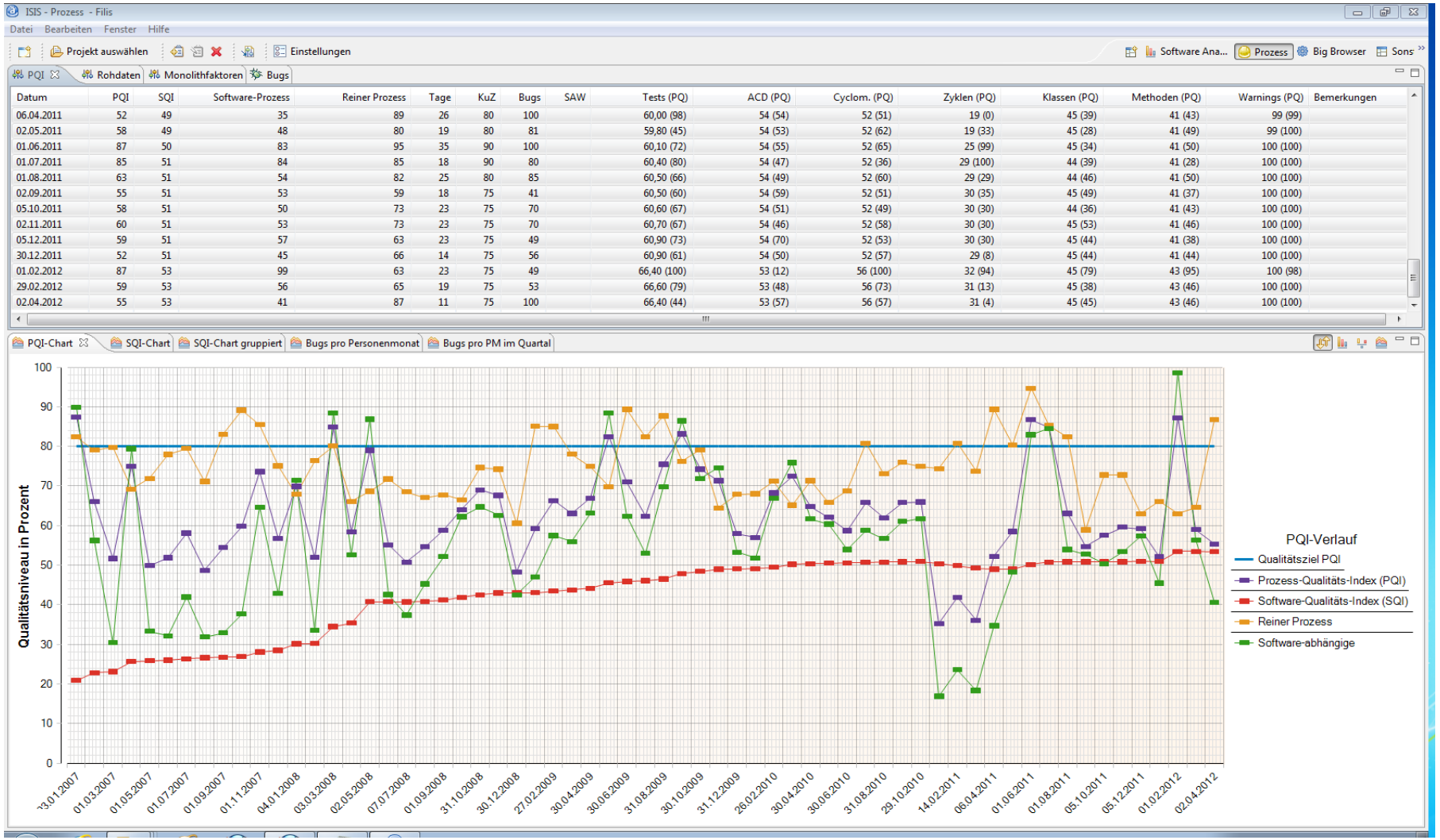
# Weniger sinnvolle Metriken

## Lines of Code (LOC)

- Problem der eindeutigen Definition
  - Kommentarzeilen
  - Zeilenumbrüche
- LOC und Funktionsumfang stehen in keinem direkten Verhältnis
  - Code-Duplizierung (Copy & Paste)
  - ineffiziente Implementierung
  - Refactoring kann Wert bei gleichem Funktionsumfang verringern
- Konsequenz: keine wirkliche Aussagekraft für Qualität

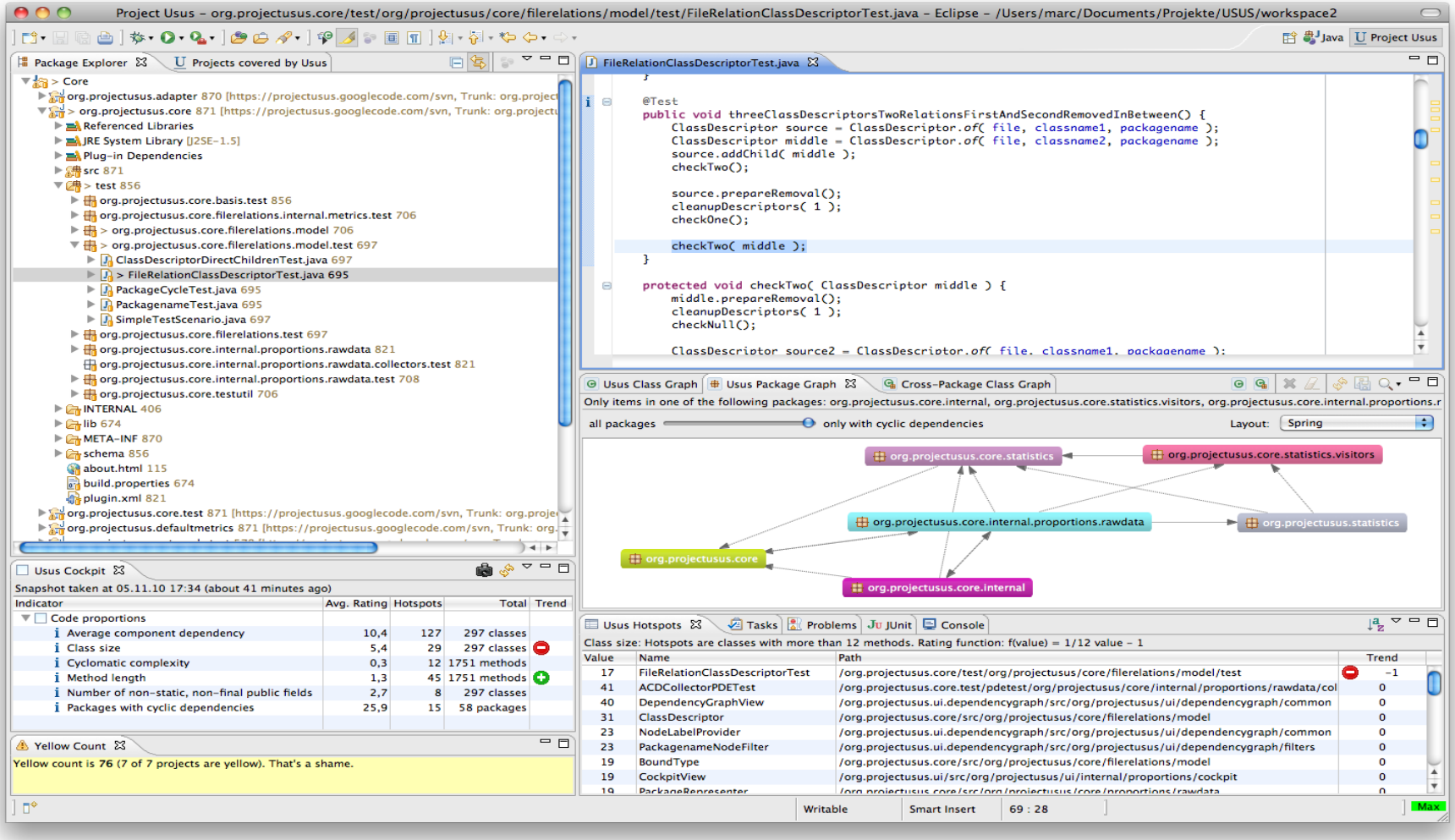
Aber: Anzahl Codezeilen pro Methode ist eine durchaus sinnvolle Metrik!

# ISIS - Es werde Licht

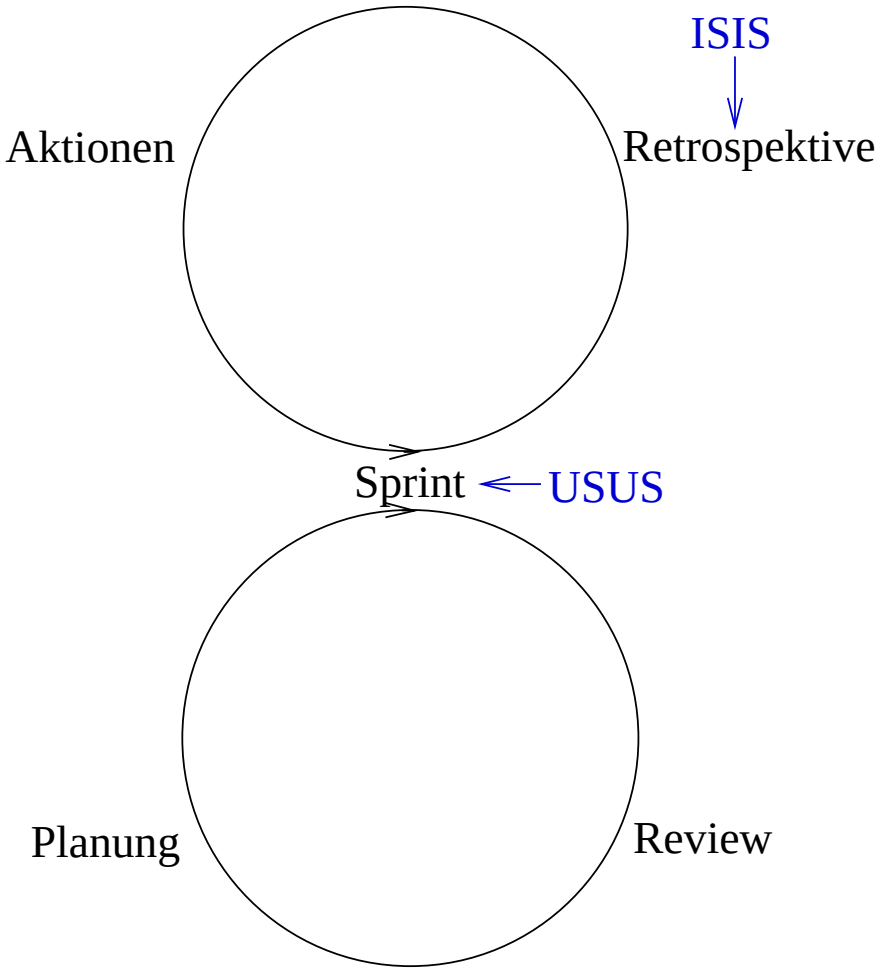


andrena OBJECTS

# USUS - Wo ist der Hund begraben?



# SCRUM - So bringt man es unter



# Integration in Prozess

## Continuous Build Server vs Integration in IDE

Continuous Build (ISIS)

- Historisierung der Änderungen
- langfristiger Erfolg wird sichtbar

Integration in IDE (USUS)

- Entwickler kann Problemstellen lokalisieren
- Entwickler sieht Erfolg der Änderung

# Integration in Prozess

## Broken Window Effect

Visualisierung der Metriken

- Erfolg messbar
- Erfolg sofort sichtbar
- ISIS: Historisierung
- USUS: sofortiger Erfolg
- Burndown-Chart: Fortschritt

# Integration in Prozess

## Maßnahmen

- TDD + PP + CI + XP einsetzen
- Refactoring gezielt einsetzen
- Code-Konventionen nutzen
- Retrospektive: Probleme adressieren

# Metriken

## Was bringen sie?

- Verstand benutzen
- Spaß beim Entwickeln
- Feedback für den Entwickler!

# USUS

## Weitere Informationen und Download

- USUS als Eclipse-Plugin
- Open Source: Eclipse Public License 1.0
- URL: http://code.google.com/p/projectusus/